

GNOME COMMAND.TXT files

GNOME COMMAND.TXT file provide a way to run GNOME in a batch mode. This is used for doing statistical runs (TAP) and for running gnome behind another interface, such as a web site, or other GUI.

These docs are a bit sparse at the moment -- please update as you find new stuff.

Using a **COMMAND.TXT** File With GNOME

The **COMMAND.TXT** file allows generation of SPLOT File series to be automated.

A **COMMAND** file is a text file with GNOME commands. The first line of a **COMMAND** file must be:

```
[GNOME COMMAND FILE]
```

Primary method:

Launching GNOME with a command file in the GNOME directory.

If there is a **COMMAND** file called **COMMAND.TXT** in the same directory as the GNOME executable, when GNOME is launched, GNOME will execute the commands in the file and then quit.

Secondary Method:

Opening/Executing command files from within GNOME

Recent versions of GNOME also support the execution of a command file through the normal "Open" file. If you open a file which is recognized as a GNOME command file, GNOME will execute the commands in the file. Note that GNOME will not automatically quit in this case. (but it will if the quite command is given at the end of the command file)

You can also start GNOME from the command line, and pass the command file name in as an argument:

```
$gnome.exe \c:\The\Path\To\The\command_file.txt
```

(note, this is a bit fragile with path handling -- it works best if you pass in the absolute path) We are working on making this approach more robust.

Using an Error Log:

Under typical GNOME usage, any Warning, Error or Note alerts would stop the model and wait for user to hit the OK button. To prevent this behavior, you can specify an Error Log file. If an error log file is specified, the text of the alert will be appended to the error log file. In the case of Warning and Note alerts, the command file continues to execute. In the case of an Error alert, execution of the command file is halted.

```
-- Set the error log file
-----
MESSAGE setField; TO model; errorLog :MyFileName.txt;
```

The commands:

In typical usage, the first part of your command file will setup the data for the scenario(s) you wish to run. There are two ways to do this:

1. use a command to open a GNOME save file
2. use a series of commands of the same type used to setup a location file.

In the second part of the command file, you would typically include one or more lines of the RUNSPILL message which runs a scenario and outputs the LEFiles.

The RUNSPILL message:

Here is an example message which includes all the required parameters:

(NOTE: in the following, long lines are continued to the next line and indented to fit on a page -- in a real file, they must be one long line.

```
MESSAGE runSpill;TO model; startRelTime 19,07,2000,17,00;
runDurationInHrs 18;timeStepInMinutes 30; numLEs 1000;
startRelPos 123.7667 W 46.21667 N;outputStepInMinutes 60;
outputFolder :Output17:;
```

The options under RUNSPILL are now available in separate parts... this need to be better integrated but should work.

Create spill in command file

```
MESSAGE createSpill; TO model; NAME MC-252; startRelTime 25,06,2010,15,00;
numLEs 1000; startRelPos 91.67 W 28.119198 N;
```

Run the model via the command file without setting a spill.

```
MESSAGE run; TO model; startTime 25,06,2010,15,00; runDurationInHrs 120;
timeStepInMinutes 15; outputStepInMinutes 60;outputFolder :MyOutput:;
```

optional parameters : note, outputPath, netcdfPath

Note that for NetCDF output, the NETCDFPATH parameter is given the same way as the OUTPUTPATH parameter.

optional parameters to support TAP extended outlook : windStartTime

Initilizing a spill from an LE File

A spill can be initialized from an LE File (as exported by GNOME in the "for GNOME Analyst" format"

```
MESSAGE createSpill; TO model; LeFilePath C:\an\example\path\test_splotsFORC
```

You can pass in the other relevant parameters, too. I think you can use an LE file for a runSpill message, too.

Optional parameters for the RUNSPILL message:

```
endRelTime, endRelPos, pollutantType, totalMass, massUnits,
windageA, windageB, persistence, z, windStartTime, runBackwards
```

Defaults:

endRelTime	startRelTime
endRelPos	startRelPos
pollutantType	Non-weathering

totalMass	numLEs
massUnits	Barrels
windageA	0.01
windageB	0.04
persistence	0.25
z	0.0 (meters)

windStartTime allows the wind to be offset from the tides to support TAP extended outlook.

pollutantType KEYWORDS:

CONSERVATIVE or "Non-Weathering"
 BUNKER or "Fuel Oil #6"
 MEDIUMCRUDE or "Medium Crude"
 IFO or "Fuel Oil #4"
 DIESEL
 JP4 or "Kerosene / Jet Fuels"
 GAS or "Gasoline"

When there are two keywords for the same oil, the first is the one used in the MOSS files and the second is the one used in the GNOME interface. You may use either one in the command file.

Note: These key words are NOT case sensitive, but you MUST include any internal spaces.

massUnits KEYWORDS:

BARRELS, GALLONS, CUBICMETERS, KILOGRAMS, METRICTONS, SHORTTONS

MESSAGE SETFIELD

The setField message can be used to change the paramaters of exiting objects. For example, the following properties of a spill can be changed:

numLEs
 startRelPos
 endRelTime
 bWantEndRelTime
 endRelPos
 bWantEndRelPosition
 pollutantType
 massUnits
 totalMass
 z
 density
 ageInHrsWhenReleased
 windageA
 windageB
 persistence

example:

```
MESSAGE SETFIELD; TO MC-252; startRelTime 01,07,2010,11,11;
    numLEs 2000; startRelPos 91.11 W 28.1111 N;
    endRelTime 01,07,2010,12,12;
    endRelPos 91.22 W 28.2222 N;
```

MESSAGE createMap

Example:

```
MESSAGE createMap; TO model; TYPE vector; NAME NGulfCoast; PATH ..\coast3.b
```

MESSAGE createMover

Examples:

```
MESSAGE createMover; TO Universal Map; TYPE Wind; NAME Offshore;
PATH winds\OFFSHOREJUNE23PM.txt; speedUnits knots;
```

```
MESSAGE createMover; TO NGulfCoast; TYPE NetCDF; NAME NCOMg;
PATH currents\NCOMg.nc;
```

Types

- Wind
- NetCDF
- CATS
- Random

Required Parameters

```
NAME NameOfMover
PATH PathToFile
```

Nowcast/forecast model data

```
MESSAGE createMover;TO GOM Map;TYPE NetCDF; NAME GOM.nc;PATH :GOM.nc; MESSAGE
createMover;TO Universal Map;TYPE NetCDF; NAME GOM.nc;PATH :GOM.nc;topFile
:GOMTopology.dat; MESSAGE createMap;TO Model;TYPE PtCur; NAME GOM Map;PATH
:GOM.nc;topFile :GOMTopology.dat;
```

The mover can either be added to a vector map or use its own ptcur map. In the latter case the single netcdf file creates both the mover and the map. If it is loaded as a current the map name will attach ' Map'. Topology file path is not necessary for regular grids.

Optional Parameters

topFile PathToFile (for a topology file for a netcdf mover)

Adding Overlays

Examples

```
MESSAGE OPEN; TO model; PATH ..\Overflights\Overflight_Mobile_2010_0625.txt;
```

To turn off the overlay

```
MESSAGE SETFIELD; TO Overflight_Mobile_2010_0625.txt; bShowOverlay False;
```

Uncertainty

```
MESSAGE setfield;TO model; INCLUDEMINREGRET true;
```

Additional Messages:

```
MESSAGE close; TO model;
MESSAGE clearWinds; TO model;
MESSAGE clearSpills; TO model;
MESSAGE reset; TO model;
MESSAGE quit; TO model;
MESSAGE save; TO model; PATH pathname;
MESSAGE startTimer; TO model;
MESSAGE stopTimer; TO model;
```

Existing command file behavior / workarounds / how to

Work around for Bug:::

Wind barbs for winds created via a command file do not show up if you give them a name in the command file. So... don't use a name when creating a wind mover.

How to set "active" via command file:::

(the name you have to use is "bActive")

```
MESSAGE SETFIELD; TO NCOMg; bActive False;
```

An example COMMAND.TXT file

```
[GNOME COMMAND FILE]

-- this would clear all maps etc in case we are not launching GNOME
MESSAGE close; TO model;

-----
-- Set the error log file
-----
MESSAGE setField; TO model; errorLog :MyFileName.txt;

-----
-- Either open a save file
-----
MESSAGE open; TO model; PATH yourRelativePathHere;

-----
-- Or use normal messages from wizard commands in location file
-----

MESSAGE setfield;TO model;timeStep 0.10;
--
MESSAGE createMap;TO model;TYPE vector; NAME CR Estuary Map;
    PATH :Columbia River:Crest.bna;
--
MESSAGE createMover;TO CR Estuary Map;TYPE Cats; NAME River Flow.CUR;
    PATH :Columbia River:River Flow.cur;
--
MESSAGE createMover;TO CR Estuary Map;TYPE Cats; NAME CR Tides.CUR;
    PATH :Columbia River:CR Tides.cur
MESSAGE setfield;TO CR Tides.CUR; scaleType constant;
    refP 123.7667 W 46.21667 N; scaleValue -1;
    timeFile :Columbia River:Shio.txt;

MESSAGE createMover;TO Universal Map;TYPE Random; NAME Diffusion;
MESSAGE setfield;TO Diffusion; coefficient 200000;uncertaintyFactor 2
```

```
-----  
-- specific setup message not typically used in LocationFiles  
-----  
MESSAGE clearWinds; TO model;  
MESSAGE createMover;TO Universal Map;TYPE Wind; NAME Variable Wind;  
    PATH :Columbia River:WindData.wnd; speedUnits knots;  
MESSAGE setfield;TO Variable Wind; windage 0.02 0.05;
```

```
-----  
-- specific messages for the TAP run  
-----  
MESSAGE runSpill;TO model; startRelTime 19,07,2000,17,00;  
    runDurationInHrs 18;timeStepInMinutes 30; numLEs 1000;  
    startRelPos 123.7667 W 46.21667 N;outputStepInMinutes 60;  
    outputFolder :TapOutput7th:;  
  
MESSAGE runSpill;TO model; pollutantType Kerosene / Jet Fuels;  
    totalMass 5000; massUnits GALLONS;startRelTime 19,08,2000,17,00;  
    endRelTime 19,08,2000,21,30; runDurationInHrs 18;timeStepInMinutes 30;  
    numLEs 1000;startRelPos 123.7667 W 46.21667 N;  
    endRelPos 123.7667 W 46.24 N;outputStepInMinutes 60;  
    outputFolder :TapOutput8th:;  
  
-----  
-- for a hindcasting run  
-----  
MESSAGE runSpill;TO model; runBackwards yes; pollutantType Kerosene / Jet Fuels;  
    totalMass 5000; massUnits GALLONS;startRelTime 19,08,2000,17,00;  
    endRelTime 19,08,2000,21,30; runDurationInHrs 18;timeStepInMinutes 30;  
    numLEs 1000;startRelPos 123.7667 W 46.21667 N;endRelPos 123.7667 W 46.24 N;  
    outputStepInMinutes 60;outputFolder :TapOutput8th:;  
  
-----  
  
-- if we wanted to quit  
--MESSAGE quit; TO model;
```

Additional commands from the GNOME code

Attachments